31. Ridiculous LLM compression techniques

Ignashin Igor, Kiselyov Ivan, Leontyeva Polina, Murzina Anastasiya
Mentor: Bulatov Aydar

AIRI Summer School 2025

Abstract

The goal of this research is to reduce the memory consumption of LLaMA while preserving model performance. Large language models like LLaMA require significant GPU memory, making them difficult to deploy on consumer hardware or cost-effective cloud instances. By analyzing and optimizing key components—such as attention mechanisms, feed-forward layers, and normalization—we aim to achieve memory-efficient inference and training without substantial accuracy loss.

1 Introduction

Current large language models face deployment challenges due to their massive memory requirements. We investigate structural compression techniques that go beyond standard parameter reduction, focusing on three key experiments:

First, we analyze attention head redundancy by isolating individual heads and measuring their functional similarity across layers. Second, we systematically evaluate layer importance through pairwise removal experiments, identifying unexpectedly replaceable components. Finally, we develop an iterative pruning method with LoRA adapters that gradually compresses the model while maintaining performance.

Using LLaMA-3.2B-Instruct as our testbed, we employ controlled ablation studies with both MMLU and Wikitext benchmarks. Our approach reveals fundamental insights about transformer architectures while providing practical compression tools. The following sections detail our methodology and findings, beginning with related work before presenting our experimental framework and results.

2 Related Work

The quest to expose the absurd inefficiencies of LLMs has taken many forms. In Your Transformer is Secretly Linear [4], the authors reveal that transformers often behave like glorified linear models in disguise, making their trillion-parameter theatrics seem even more ridiculous. Meanwhile, [1] drop another truth bomb in The Unreasonable Ineffectiveness of the Deeper Layers, showing that many deep layers in LLMs are about as useful as a screen door on a submarine. These works join our crusade against over-engineering by proving that LLMs are often stupidly redundant—whether through linearity masquerading as complexity or layers that contribute nothing but FLOPs.

3 Method

.

3.1 Defining similar heads

We extracted and analyzed the representations of attention heads across different layers using a forward pass on the MMLU dataset. By computing pairwise cosine similarities between attention head representations, we identified redundant heads that could be pruned or merged without significant performance degradation. We analyzed LLaMA-3.2B (28 layers, 24 heads/layer) using 20 diverse examples from MMLU's dev split.

This approach combined: 1) **Head isolation technique**: the head isolation technique selectively evaluates individual attention heads by suppressing all other heads in a layer during forward propagation. This produces pure head representations uncontaminated by cross-head interactions.

$$\text{Attention}_i(Q', K', V') = \text{softmax}\left(\frac{Q'_i K'^T_i}{\sqrt{d_k}}\right) V'_i \tag{1}$$

2) **Representation Extraction and Similarity Analysis:** for each head, we extracted its output representation across 20 diverse examples from the MMLU dev split. Pairwise cosine similarities were computed between all heads (within and across layers) to measure functional redundancy.

$$\text{similarity} = \frac{x_1 \cdot x_2}{\max(\|x_1\|_2 \cdot \|x_2\|_2, \epsilon)}. \tag{2}$$

Similarity matrices were visualized as heatmaps to identify clusters of highly correlated heads.

Looking at the similarity matrices, we observed the following key patterns:

1) Layer 0 shows minimal similarity: The heads in the first layer (Layer 0) exhibit almost no functional redundancy with other heads, suggesting they capture diverse, low-level features unique to their position in the network. This aligns with expectations that early layers process basic patterns before higher layers combine them.

2) Similarity increases with depth: As we move to deeper layers (especially Layers 13-14 and 26-27), pairwise similarities between heads grow significantly. This indicates:

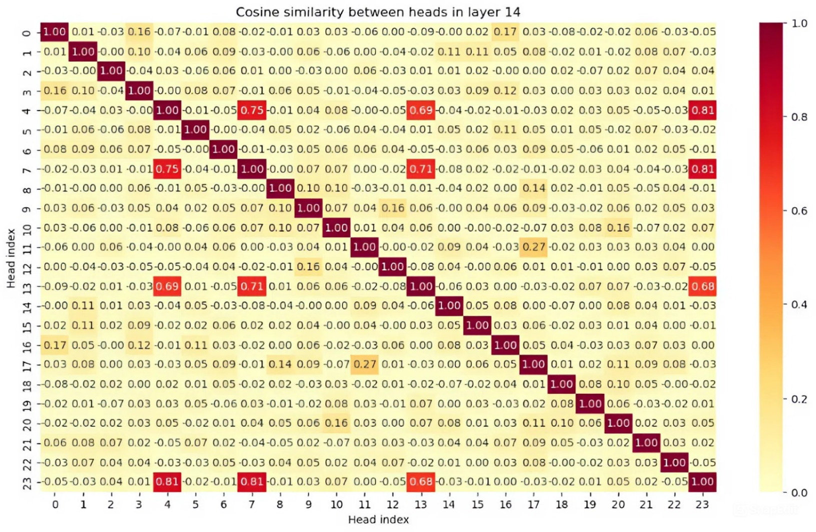

Figure 1: Attention head similarity patterns across LLaMA-3.2B layers. The heatmap visualizes pairwise cosine similarities between head representations, revealing increasing redundancy from Layer 0 (low similarity) to deeper layers (higher similarity). Darker squares indicate stronger similarity between heads, with notable block patterns emerging in middle (Layers 13-14) and final layers (26-27). The diagonal structure suggests layer-specific functionality, while off-diagonal patterns reveal cross-layer relationships.

3) Higher layers develop more specialized, task-oriented representations. Redundancy emerges as the model consolidates features for downstream tasks.

4) Adjacent layers share functionality: The strongest similarities often occur between heads in consecutive layers (e.g., Layer 13 Layer 14), suggesting gradual feature refinement. Final layers show task-specific convergence: The last layers (26-27) demonstrate the highest similarity scores, particularly for heads processing the same examples, implying they converge on similar high-level representations to solve the task.

These findings suggest distinct optimization strategies should be applied at different network depths. The unique, low-level feature extraction in early layers (particularly Layer 0) warrants preserving all heads, as their diversity appears fundamental to the model's processing pipeline. However, the increasing redundancy in middle and later layers presents clear opportunities for optimization.

3.2 Systematic Double-Layer Removal Analysis

To investigate the contribution, redundancy and replaceability of individual layers within the transformer architecture, we conducted a layer-pruning experiment. The core idea is to measure the degradation in model performance when layers are removed entirely. This approach helps identify which layers are critical and which layers might be functionally redundant.

The procedure was as follows:

1. **Model and Baseline:** We used the `Llama-3.2-3B-Instruct` model. First, we established a baseline performance metric by calculating the perplexity of the original, unmodified model on the test set of the `wikitext` dataset.

2. **Iterative Pruning:** We iterated through all unique pairs of layers (i, j) in the model, where $i < j$. To focus the analysis on the main body of the network, we excluded the earliest layers (0, 1, 2) and the final layer from the removal process, as these have shown to be critically important, completely breaking the model if removed (they perform initial feature embedding and final output processing).

3. **Model Modification:** We then programmatically removed both the i-th and j-th transformer layers.

4. **Performance Evaluation:** The perplexity of each pruned model was evaluated on the same `wikitext` dataset. This allowed for a direct comparison of performance degradation relative to the baseline.

5. **Result Aggregation:** The resulting perplexity score for each removed pair (i, j) provides a comprehensive map of the model's sensitivity to the removal of any two layers and allows to roughly estimate relative importance of each layer to each layer.

3.3 Layer Interchangeability Analysis

To further probe the functional roles of different layers, we conducted a layer replacement experiment. Instead of merely removing layers, this analysis evaluates how well one layer's functionality can be substituted by another's. This helps to map out functional similarities across the entire network depth. Since we are going to use multiple copies of a single layer, that won't require any addional memory, so layer replacement could potentially be a good method of model compression.

The methodology was as follows:

1. **Systematic Replacement:** We iterated through all possible ordered pairs of layers (i, j), where i is the index of the layer to be replaced (the "recipient") and j is the index of the layer to be copied (the "donor").
2. **Model Modification:** The parameters of the recipient layer 'i' were removed and instead the parameters from the donor layer 'j' were used. The rest of the model architecture remained unchanged. The case where $i = j$ is equivalent to the original, the model stays unmodified.
3. **Performance Evaluation:** The perplexity of each modified model was evaluated on the `wikitext` test set, using the same procedure as other experiments. This provided a measure of how successfully layer j could perform the role of layer i.

3.4 Centered Kernel Alignment (CKA)

To quantitatively measure the representational similarity between any two layers, we employed Centered Kernel Alignment (CKA) [2]. Given the activation matrices $X \in \mathbb{R}^{m \times d_1}$ and $Y \in \mathbb{R}^{m \times d_2}$ from two layers for m different inputs, CKA computes a normalized similarity score. First, the Gram matrices $K = XX^\top$ and $L = YY^\top$ are computed. After centering them with the matrix $H = I_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$, the CKA index is given by:

$$\mathrm{CKA}(K, L) = \frac{\mathrm{HSIC}(K, L)}{\sqrt{\mathrm{HSIC}(K, K) \cdot \mathrm{HSIC}(L, L)}} \tag{3}$$

where $\mathrm{HSIC}(K, L) = \frac{1}{(m-1)^2}\mathrm{tr}(HKHL)$ is the Hilbert-Schmidt Independence Criterion. The resulting score ranges from 0 (dissimilar) to 1 (identical representations). This was done to analyze how activations affect layers interchangeability.

3.5 Layer Interpolation

To analyze the functional space between two layers, we performed layer interpolation. This method involves creating a new layer whose parameters are a weighted average of the parameters from a "donor" layer j and a "recipient" layer i. The parameters for the new, interpolated layer (θ_{new}) are calculated as:

$$\theta_{\text{new}} = \alpha \cdot \theta_j + (1 - \alpha) \cdot \theta_i \tag{4}$$

where θ_i and θ_j are the weights of the respective layers, and $\alpha \in [0, 1]$ is the interpolation coefficient. By evaluating model performance (perplexity) across different values of α, we can assess the functional compatibility of two layers.

3.6 LLama Decoder Layer Hidden State Metric Analysis

This subsection presents a quantitative analysis of hidden state transformations within the LLaMA decoder layer. By examining how hidden states evolve between key components—input layer normalization (LayerNorm), self-attention, post-attention LayerNorm, and MLP—we characterize the layer's information processing dynamics. The analysis focuses on four key metrics computed between consecutive components:

Mean Difference $\mathbb{E}(\Delta h^{(l)}) = \frac{1}{N \times d} \sum_{i=1}^{d} \sum_{n=1}^{N} (h_{i,n}^{(l)} - h_{i,n}^{(l+1)})$ measures the average change in hidden state values, indicating the direction and magnitude of transformations. The near-zero fluctuation indicates balanced positive/negative updates or residual connection dominance. We are going to detect saturation in attention heads and identify gradient flow issues (vanishing updates in deep layers).

Mean Absolute Difference $\mathbb{E}(|\Delta h^{(l)}|) = \frac{1}{N \times d} \sum_{i=1}^{d} \sum_{n=1}^{N} |h_{i,n}^{(l)} - h_{i,n}^{(l+1)}|$ captures the intensity of changes, regardless of direction. The high values of MAD Metric is a sign of aggressive feature transformations. Here we compare relative impact of different components.

Standard Deviation of Differences $\sigma^{(l)} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{d} \sum_{n=1}^{d} (\Delta h_{i,n}^{(l)} - \mu_n^{(l)}) \right)}$ quantifies the variability in transformations across dimensions. It's known that Attention layers typically show higher σ than MLPs (sparse updates).

Cosine Similarity $\mathbb{E}(\cos \theta^{(l)}) = \frac{1}{N} \sum_{i=1}^{N} \frac{h_i^{(l)} h_i^{(l+1)}}{\|h_i^{(l)}\| \times \|h_i^{(l+1)}\|}$ Evaluates angular similarity between hidden states, revealing directional consistency.

4 Experimental Setup

Our primary model for the compression experiments was Llama-3.2-3B-Instruct. To evaluate the impact of our compression techniques, we used perplexity as the key performance metric. Perplexity was calculated on the test split of the `wikitext` dataset [3], a standard benchmark for language modeling quality. The baseline perplexity of the unmodified model was established first, and all subsequent modified models were evaluated against this benchmark.

4.1 Experiment Report: Pruning-Healing Llama

This experiment compares two strategies for reducing and adapting the LLaMA 3.2 Instruct 3B model: a baseline one-step pruning method and a proposed Iterative LoRA Healing approach.

Setup:

- **Base model:** LLaMA 3.2 Instruct 3B
- **Training data:** `wikitext-2-raw-v1` (0.5k iterations per step)
- **Evaluation data:** `wikitext-103-raw-v1` (1k iterations)

Baseline approach:

- Remove 4 transformer layers simultaneously.
- Apply a LoRA adapter only to the last MLP layer.
- Fine-tune the model for 3000 iterations.

Iterative LoRA Healing:

- Sequentially remove layer k.
- Shift its responsibility to layer $k+1$, and insert a LoRA adapter into $k+1$.
- Fine-tune layer $k+1$ for 500 iterations.
- Repeat this process for 5 consecutive layers.
- Total iterations: $5 \times 500 = 2500$

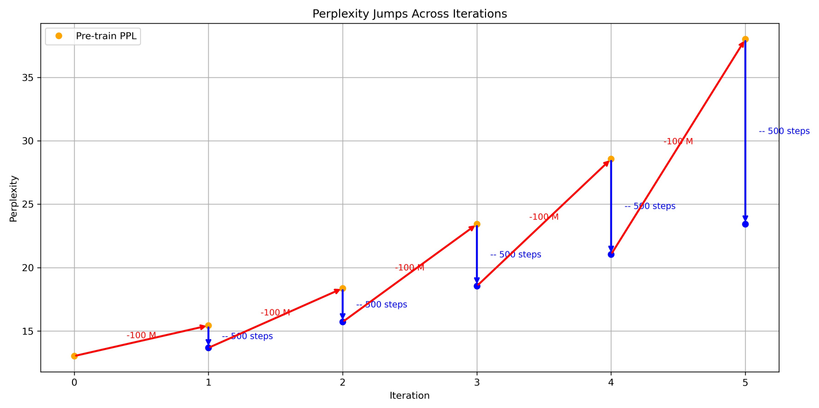

Figure 2: Iterative pruning and LoRA fine-tuning results

LoRA fine-tuning as knowledge distillation: Originally, the model factorizes the conditional likelihood as

$$\log p_k(x_{k+1} \mid x_k) + \log p_{k-1}(x_k \mid x_{k-1}),$$

where each layer models dependencies between successive hidden states. After removing layer k, the conditional becomes

$$\log p_k(x_{k+1} \mid x_{k-1}),$$

forcing layer k to directly process inputs from x_{k-1} without the intermediate x_k. Fine-tuning a LoRA adapter on layer k helps it *distill* and absorb the missing information originally captured by p_{k-1}, thus preserving the overall conditional structure despite the skipped layer.

Process details: Each time a layer is removed, the following layer takes its place. This layer is enhanced with a LoRA adapter and trained to absorb the previous layer's function. As this process is repeated, the model gradually shrinks, while local fine-tuning keeps it stable. The result is a temporary spike in perplexity after each pruning, followed by recovery through adaptation. **Key observations:**

- **Perplexity trends:** Each layer removal causes a brief increase in perplexity, which is then reduced through LoRA fine-tuning.
- **Performance:** The final model obtained through Iterative LoRA Healing outperforms the baseline in perplexity, despite similar training budget.
- **Robustness:** This approach leads to a more stable and less degraded model, thanks to step-by-step adaptation of neighboring blocks.

Note on evaluation: Perplexity, though imperfect for downstream tasks, is effective for detecting structural degradation from architectural changes.

Conclusion: Iterative LoRA Healing offers a more robust and natural alternative to direct pruning and minimal adaptation. By sequentially shifting capacity upward in the model and locally restoring performance via LoRA adapters, we achieve higher final quality at comparable compute. Unlike the approach in [4], we do not replace removed decoder blocks with simple linear projections. Moreover, unlike [1], we do not prune consecutive layers. Instead, we remove every other layer and apply a LoRA adapter to the next one, allowing it to compensate for the functional gap introduced by the removed layer. This is a more natural mechanism, given the high degree of flexibility in decoder blocks and the observed similarity between neighboring attention heads. This technique enables graceful model reduction and could be useful in low-resource adaptation settings or efficient distillation pipelines.

5 Results

5.1 Layer Hidden State Metrics

Evolution of hidden state transformation metrics across LLaMA decoder layers, revealing two dominant patterns (Fig. 3):

1. Residual connection dominance emerges at $\sim 3/4$ depth (layers 19-23), characterized by near-zero mean difference $\mathbb{E}(\Delta h) \approx 0$ and high cosine similarity $\mathbb{E}(\cos\theta) > 0.93$, indicating skip connections govern signal propagation;

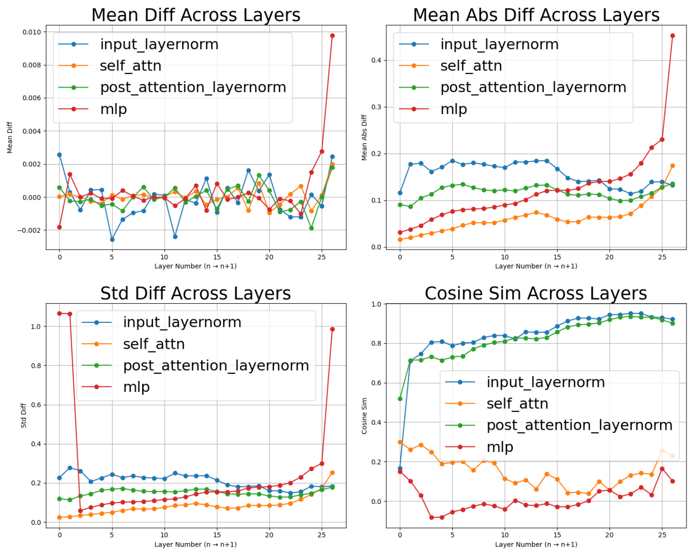

Figure 3: Layer-wise evolution of hidden state transformation metrics (mean difference, absolute difference, standard deviation, and cosine similarity) across all LLaMA decoder layers

2. MLP-driven feature recombination dominates final layers (27-28), evidenced by peak mean absolute differences $E(|\Delta h|) > 0.2$ that is significantly more previous layers the transformation Metric.

5.2 Layer Removal

The results of our double-layer removal experiment are visualized in the perplexity heatmap in Figure **??**. The color of each cell (i, j) represents the model's perplexity after removing both layer i and layer j.

The heatmap reveals three patterns:

1. **High Redundancy of Some Layers:** Some of the layers could be removed without hurting the performance significantly.
2. **A Less Critical Middle Block:** A large, square-like region is visible in the middle layers of the network (approximately layers 7 through 13). Removing pairs of layers from this block is less damaging to the model's performance than removing layers from other regions.
3. **Critical Early and Late Layers:** The early and late layers perform more unique and critical functions that cannot be easily compensated for, and their removal is highly detrimental, rendering the model absolutely useless.

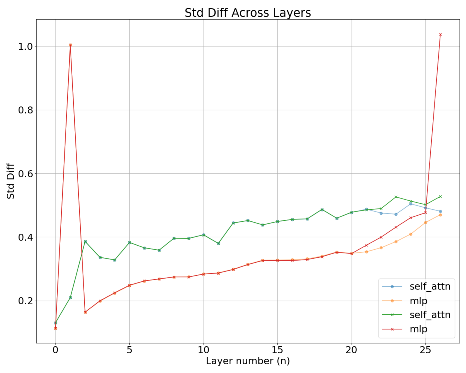

Figure 4: Caption

5.3 Layer Interchangeability

The layer replacement experiment, summarized in Figure **??**, provides a detailed map of functional similarity. In this heatmap, the y-axis represents the layer being replaced and the x-axis represents the layer whose weights are copied. Brighter reddish colors signify a successful replacement (low perplexity).

Key observations include:

- **Strong Positional Dependence:** The map is asymmetric. The top-right quadrant is dark, showing that replacing early-stage layers with late-stage layers results in catastrophic failure. This highlights that layers are specialized for their position in the network's processing hierarchy, late layers require data processed in previous layers.
- **Universal Donors and Recipients:** The heatmap shows clear vertical and horizontal bands. A bright vertical column indicates a "universal donor" layer whose function is general enough to be transplanted into many other positions. A horizontal row indicates a "versatile recipient" layer whose role is generic enough to be fulfilled by many other layers.
- **Local Interchangeability:** Despite being asymmetric in general, the heatmap has symmetry in neighborhood of the main diagonal, suggesting that layers that lay close to each other may often be interchangeable.

5.4 CKA Confirms Representational Similarity Blocks

To investigate whether functional similarity corresponds to representational similarity, we used Centered Kernel Alignment (CKA). The resulting heatmap is shown in Figure **??**. Brighter colors indicate a higher CKA score and thus more similar layer activations.

5.5 Layer Interpolation Probes Functional Compatibility

Finally, the layer interpolation experiment (Figure **??**) examines the "functional path" between layers. The plots show perplexity as the weights of a layer are linearly interpolated between its original state ($\alpha = 0$) and the weights of a donor layer ($\alpha = 1$).

6 Conclusion

References

[1] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers, 2024. *URL https://arxiv. org/abs/2403.17887*.

[2] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 3519–3529. PMLR, 2019.

[3] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[4] Anton Razzhigaev, Matvey Mikhalchuk, Elizaveta Goncharova, Nikolai Gerasimenko, Ivan Oseledets, Denis Dimitrov, and Andrey Kuznetsov. Your transformer is secretly linear. *arXiv preprint arXiv:2405.12250*, 2024.

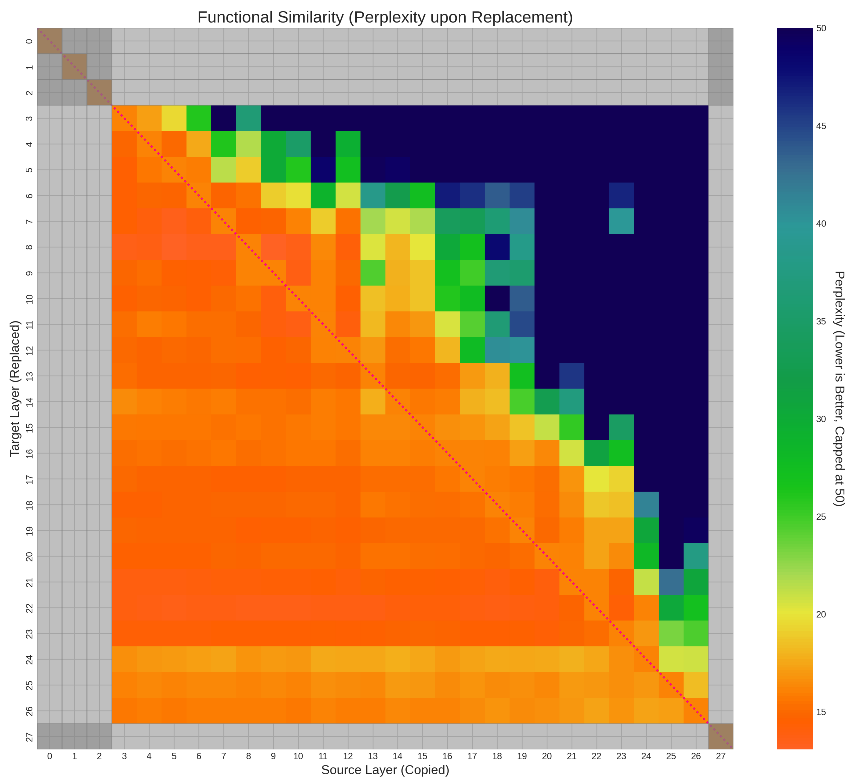

Figure 5: Layer interchangeability analysis

7 Appendix

Graphical representations of experiment results

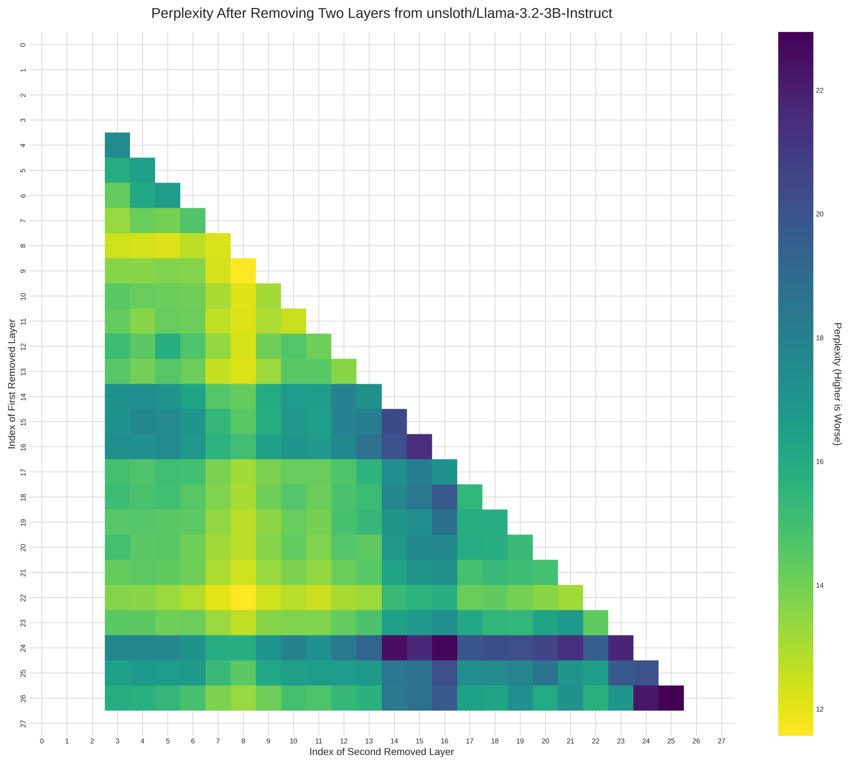

Figure 6: Pairs of layers removal

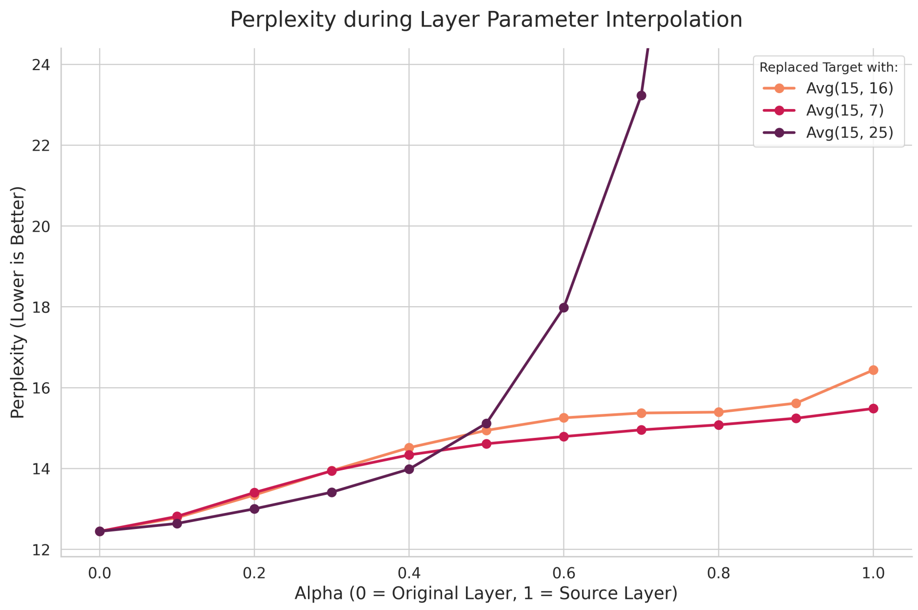

Figure 7: How does perplexity change as we interpolate layers weights between original values and values from best, median, and worst suitable layer?

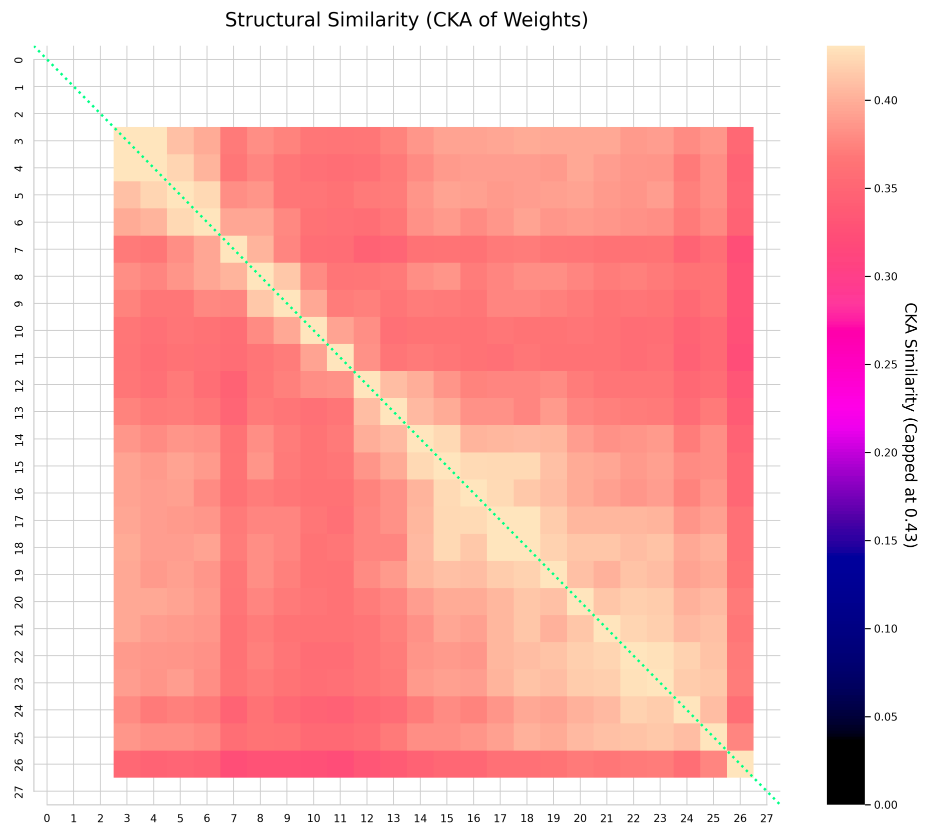

Figure 8: How similar are the activations of pairs of layers based on the CKA metric?

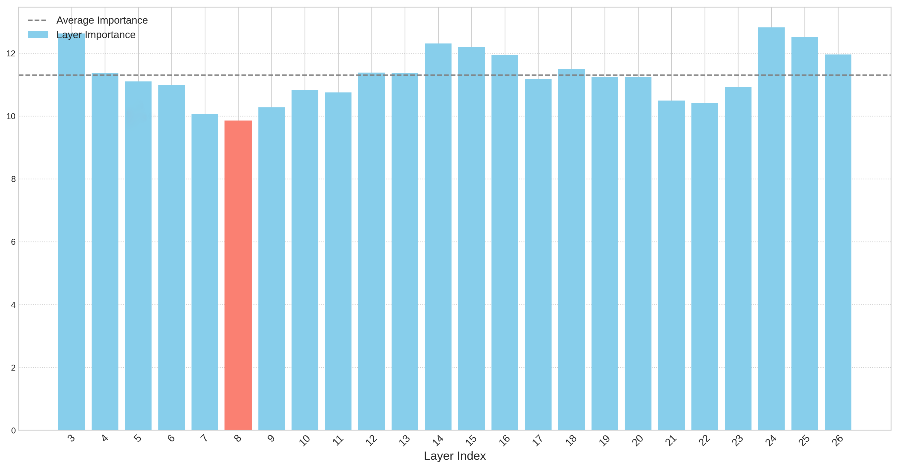

Figure 9: Layer importance (red layer has least impact on perplexity)